AD-A100 389 NAVAL RESEARCH LAB WASHINGTON DC F/G 17/2
ADDITIVE AND MULTIPLICATIVE DISTORTION SIMULATOR FOR NON-REAL T--ETC(U)
JUN 81 W M JEWETT, R COLE
UNCLASSIFIED NRL-MR-4532 NL

1 OF 1
AD
A100389

END
DATE
FILMED
7-81
DTIC

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
NRL Memorandum Report 4532

2. GOVT ACCESSION NO.
AD-A100 389

3. RECIPIENT'S CATALOG NUMBER
14 NRL-MR-4532

4. TITLE (and Subtitle)
ADDITIVE & MULTIPLICATIVE DISTORTION SIMULATOR FOR NON-REAL TIME TESTING OF HF MODEMS

5. TYPE OF REPORT & PERIOD COVERED
Interim report on a continuing NRL problem.

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
W. M/Jewett ~~and~~ R./Cole, Jr.

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Research Laboratory
Washington, DC 20375

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
33401N; X0736-CC;
75-0126-0-1

11. CONTROLLING OFFICE NAME AND ADDRESS
11 11 Jun 81

12. REPORT DATE
June 11, 1981

13. NUMBER OF PAGES
~~52~~ 12 53

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
16 X0736
17 X0736-CC

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
HF modems
Channel simulation
ANDVT

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
A non-real time FORTRAN program has been developed to provide a simple, standardized means of applying uniform tests to various HF modem signals. The program stresses the modem signal in the same manner as would be experienced by operation on fading channels with tonal interference. Provisions are made for inserting various transmission filters and for operational scenarios that stress the doppler acquisition and tracking capabilities of a modem.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

ADDITIVE AND MULTIPLICATIVE DISTORTION

SIMULATOR FOR NON-REAL TIME TESTING

OF HF MODEMS

1.0 Introduction

This report describes an hf channel simulator which includes the propagation path, the frequency response of the rf equipment, inband interference from other users of a common channel, and the operational conditions. The operational conditions are related primarily to doppler problems incurred with high velocity aircraft.

This is an off-line, non-real time test capability using sampled data input from a modulator and providing sampled data output to a demodulator. Test conditions are established in a table of variables, permitting a flexible means of selecting combinations of conditions. It is the purpose of this report to describe the capabilities of this simulation program.

2.0 Simulation Program

2.1 Types of Stress

A FORTRAN program (Appendix A) has been developed which permits the application of a set of selectable stress conditions to a digitized modem line signal. The program is independent of the modem signal design, depending only on the sampling rate used to digitize the input signal.

The types of stress for which test conditions may be specified include:

- Gaussian noise level
- frequency translation error
- multipath delay with fixed doppler offset
- varying doppler due to operational scenario
- multitone interference

Manuscript submitted April 9, 1981.

- random noise pulse interference
- equipment filter characteristics
- AGC characteristics.

The program uses independent subroutines for introducing each stress condition. The parameters for describing the test conditions are contained in a two-dimensional (10 x 10) floating point array which is accessed by the subroutines. Table 1 is a list of all the variables in that array with a description of their use. Appendix B is a FORTRAN program used to generate the table, and Appendix C is a program to modify an existing table.

Figure 1 is a block diagram showing the order in which the stress functions may be performed. Any of the functions may be readily bypassed. Noise and interference are added after introducing the multipath and doppler distortions. Filtering is divided into a transmit and a receive filter, making it possible to filter the desired signal twice, but the interference only once. Table 2 is a list of the subroutines and their use.

2.2 Input/Output

The test program operates from a sampled digitized input of the modulator output signal. The input may be from either magnetic tape or a disk file. In either case, each sample is quantized to 12 bits and left justified in a 16 bit word. The data are stored in blocks of 1000 samples. The magnetic tape is a standard nine-track, 800 byte per inch recording system.

Each integer input sample is converted to floating point and normalized to a value corresponding to zero dBm across 600 ohms. Normalization of the input data permits all interference levels to be specified relative to zero dBm.

The sampled data is sequenced through the various subroutines in blocks of equal size, whose length is specified in the table of variables. It is

Table 1 — Identity of variables in control table

I \ J	1	2	3	4	5	6	7	3	9	10
1	SAMP	GAUSS	PATH DELAY	PATH AMP	NCW	NFSK	NPSK	F1SAW	START	PROB
2	NBD	AGC GAIN			F1CW	F1MK	F1PSK	F2SAW	DOPX	AMPPUL
3	NPATH	NTAPE			FXCW	SHIFT	FXPSK	RSAW	RATE	PULDUR
4	NFADE	MTAPE			AMPCW	FXFSK	AMPPSK	AMPSAW	DOPLOW	
5	KDOPP	NAGC				AMPFSK			DOPHGH	
6	KINTF	AGC DEC							TIMEY	
7	KPULSE	AGC MIN							XDIST	
8	KTXFIL								VEL	
9	KRXFIL								RF	
10	KAGC								DOPP	

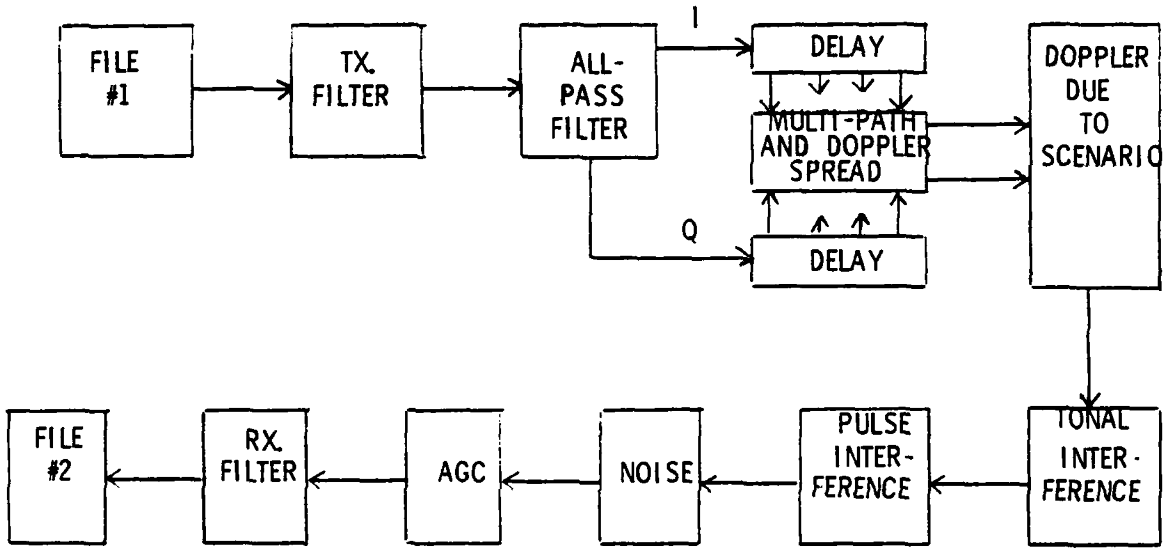

Fig. 1 — Non-real-time stress program

Table 2 — Identity of subroutines

1. INPUTX
   Read in NBD sample of data from either magnetic tape or a disk file, representing modulator output signal.

2. Tx FILT
   Transmitter filter; selection of one of five algorithms to filter input data.

3. ALPASX
   All-pass filter NBD samples of data to generate inphase and quadrature components.

4. DELAYX
   Combine output of N propagation paths for different delays.

5. FREQX
   Frequency translate composite signal by a fixed or variable amount.

6. DOPVAR
   Update doppler frequency in accordance with one of three operational scenarios.

7. INTERF
   Add non-fading tonal interference to fading signal. Interference may be CW, FSK, DPSK, and/or Swept Tone.

8. PULSE
   Add wideband Gaussian noise pulse to received signal.

9. NOISEX
   Add Gaussian noise to received signal.

10. RxFILT
    Receiver filter, selection of one of five algorithms to filter output signal plus noise.

11. AGC
    Receiver automatic gain control.

12. STOREX
    Store NBD samples of data in integer record for transfer to either magnetic tape or disk file.

13. MTIN
    Read block of 1000 samples from magnetic tape.

14. MTSET
    Set MTOUT for D/A operation

15. MTOUT
    Write block of 1000 samples on magnetic tape.

convenient to make this block size equal to the number of samples in a baud of the modulated tonal interference. In the absence of such an interference, the block size may be any value up to 100 samples.

After all signal processing, the normalization is removed and the distorted signal is converted to integer and stored on the output medium in the same format as the input signal. The output medium may be either magnetic tape or a disk file.

2.3 Transmission Equipment Filters

Provisions are made to select one of five forms of a simple non-recursive bandpass filter. Characteristics of the five filters are completely specified within the filter subroutines. All five filters are symmetrical about the mid-band frequency, which is one-fourth of the sampling rate. The filters are defined by the following equations:

1. $y_0 = 0.5\ X_0 - 0.5\ X_{-2}$
2. $y_0 = 0.25\ X_0 - 0.5\ X_{-2} + 0.25\ X_{-4}$
3. $y_0 = 0.6\ X_0 - 0.5\ X_{-2} - 0.1\ X_{-4}$
4. $y_0 = 0.36\ X_0 - 0.6\ X_{-2} + 0.13\ X_{-4} + 0.1\ X_{-6} + 0.01\ X_{-8}$
5. $y_0 = 0.1296\ X_0 - 0.432\ X_{-2} + 0.453\ X_{-4} - 0.084\ X_{-6}$
$-0.0959\ X_{-8} + 0.014\ X_{-10} + 0.0127\ X_{-12}$
$+0.002\ X_{-14} + 0.0001\ X_{-16}$

Table 3 gives the amplitude and phase response of the five filters. Filter one is the simplest form of a non-recursive bandpass filter. It has a 3 dB bandwidth of 0.5 W, where W is the Nyquist frequency. The phase response is linear. Filter two is a cascade of two of the number one filters. The result is a 6 dB bandwidth of 0.5 W and a 3 dB bandwidth of 0.36 W. The phase response is still linear.

Table 3 — Amplitude and phase response of bandpass filters

FREQ.	Filter #1		Filter #2		Filter #3		Filter #4		Filter #5	
	AMP	PHASE	AMP	PHASE	AMP	PHASE	AMP	PHASE	AMP	PHASE
.00	.0000	90.00	.0000	90.00	.0000	0.00	.0000	180.00	.0005	180.00
.05	.1564	81.00	.0245	162.00	.2177	78.45	.0474	156.91	.0017	-52.69
.10	.3090	72.00	.0955	144.00	.4224	67.07	.1784	134.13	.0312	-92.26
.15	.4540	63.00	.2061	126.00	.6027	56.00	.3632	112.00	.1312	-136.12
.20	.5878	54.00	.3455	108.00	.7500	45.43	.5626	90.85	.3160	-178.33
.25	.7071	45.00	.5000	90.00	.8602	35.54	.7400	71.03	.5472	142.12
.30	.8090	36.00	.6545	72.00	.9336	26.51	.8716	53.03	.7593	106.02
.35	.8910	27.00	.7939	54.00	.9752	18.50	.9510	37.00	.9036	73.97
.40	.9511	18.00	.9045	36.00	.9937	11.54	.9874	23.08	.9744	46.14
.45	.9877	9.00	.9755	18.00	.9992	5.50	.9984	11.00	.9963	21.99
.50	1.0000	0.00	1.0000	0.00	1.0000	0.00	1.0000	0.00	.9995	0.00
.55	.9877	-9.00	.9755	-18.00	.9992	-5.50	.9984	-11.00	.9963	-21.99
.60	.9511	-18.00	.9045	-36.00	.9937	-11.54	.9874	-23.08	.9744	-46.14
.65	.8910	-27.00	.7939	-54.00	.9752	-18.50	.9510	-37.00	.9033	-73.97
.70	.8090	-36.00	.6545	-72.00	.9336	-26.51	.8716	-53.03	.7593	-106.02
.75	.7071	-45.00	.5000	-90.00	.8602	-35.54	.7400	-71.08	.5472	-142.12
.80	.5878	-54.00	.3455	-108.00	.7500	-45.43	.5626	-90.85	.3160	178.33
.85	.4540	-63.00	.2061	-126.00	.6027	-56.00	.3632	-112.00	.1312	136.12
.90	.3090	-72.00	.0955	-144.00	.4227	-67.07	.1784	-134.13	.0312	92.26
.95	.1564	-81.00	.0245	-162.00	.2177	-78.45	.0474	-156.91	.0017	52.69
1.00	.0000	-90.00	.0000	-90.00	.0000	-79.65	.0000	-180.00	.0005	-180.00

Filter three has a flat top amplitude response with a 3 dB bandwidth of 0.63 W. The phase response is still quite linear. This filter is equivalent to a parallel implementation of two filters, $y_0 = 0.5\ X_0 - 0.5\ X_{-2}$ and $y_0 = 0.1\ X_0 - 0.1\ X_{-4}$

Filter four is a cascade of two of the number three filters. The result is a 6 dB bandwidth of 0.63 W and a 3 dB bandwidth of 0.51 W. Filter five is a cascade of four of the number three filters. The result is a 12 dB bandwidth of 0.63 W, a 6 dB bandwidth of 0.51 W, and a 3 dB bandwidth of 0.43 W.

For convenience, a separate filter subroutine is designated for the transmitter and the receiver filter functions. Two values in the table of variables control which filter algorithms are selected. These values may be zero through five, with zero representing no filtering. It is not necessary that the same filter characteristics be used for both the transmit and receive filters.

2.4 Analytic Signal

The envelope of the real signal is the absolute value of the signal preenvelope (analytic signal),

$$f_a(t) = f(t) + j\hat{f}(t)$$

where

$f_a(t)$ is the analytic signal as a function of time

$f(t)$ is the real signal

$\hat{f}(t)$ is the Hilbert transform of $f(t)$

The benefit of the analytic signal is that it provides a means to frequency shift the transmitted signal in the time domain. This may be performed by multiplying the analytic signal by a complex exponential.

A digital all-pass filter is used to generate an approximation to the ideal Hilbert transform pair. The input to the all-pass filter is the real signal. The two outputs are the in-phase and quadrature components of the analytic signal, that have nearly equal magnitude and are approximately 90 degrees apart. An alternate method of generating an approximation to the ideal Hilbert transform is by means of a non-recursive discrete Hilbert transform. Such designs can yield an ideal phase response, but there are fluctuations in the magnitude response, depending on the number of terms used.

2.5 Multipath

Program inputs from the table of variables permit the user to specify the number of propagation paths, their relative delays, and their relative amplitudes. The mean frequency offset and the doppler spread of each path are determined directly from an input data file. One means of generating that data is by the FORTRAN program in Appendix D. That program generates the complex function representing the fading on each path at a sampling rate equal to 1/100 of the modem sampling rate. Each value is equal to the instantaneous amplitude of the sum of several sinusoids with small differences in frequency.

$$E(t) = \sum_{n=1}^{N} \cos(w + w_n)\ t + j \sin\ (w + w_n)$$

where

> w is the mean frequency offset of the path; w_n is one of N frequencies selected to be approximately (but not exactly) uniformly distributed over the fading bandwidth

The values for each path are interleaved and stored in blocks of 1024 samples. These values are generated at 1/100 of the modem sampling rate in order to lessen the storage requirements. This channel sampling rate is well above the

minimum rate requirements for the fading channel. The simulation program interpolates between input values to obtain values at the modem sampling rate. Linear interpolation is used.

The program in Appendix D for generating sampled values of the fading on the propagation paths uses N equal amplitude complex sinusoids, where N is a variable. The frequency of each sinusoid is chosen randomly with the restriction that there is one assignment in each of N equal width increments over the total range specified for the frequency spread due to the propagation path. The inputs to that program are the number of tones per paths, the number of paths, the frequency spread for each path, and the mean frequency offset for each path. The relative amplitude of the different paths is a variable in the main simulation program.

The complex gain values are generated by combining N sinusoids, each of which have a rms amplitude of 1.0. Thus, the rms amplitude of the composite signal is equal to $\sqrt{N}$. To keep the rms amplitude of the composite signal independent of the number of sinusoids used to generate it, the complex components are divided by $\sqrt{N}$. Thus, the normalized rms amplitude of the composite signal is always 1.0.

Figure 2 is a plot of the cumulative distribution of the instantaneous amplitude of the fading signal for a one-path channel model with a spread of 1.0 Hz. These data were generated by combining 20 sinusoids. The frequencies of the 20 sinusoids are listed in Table 4. The modem sampling rate was specified as 7200 Hz, thus the sample rate for the path data was 72 Hz. Figure 3 is a plot of the median fade duration vs. doppler spread. A fade was defined as the period of time for which the instantaneous amplitude was below a threshold. More exactly, the beginning of a fade was defined as when

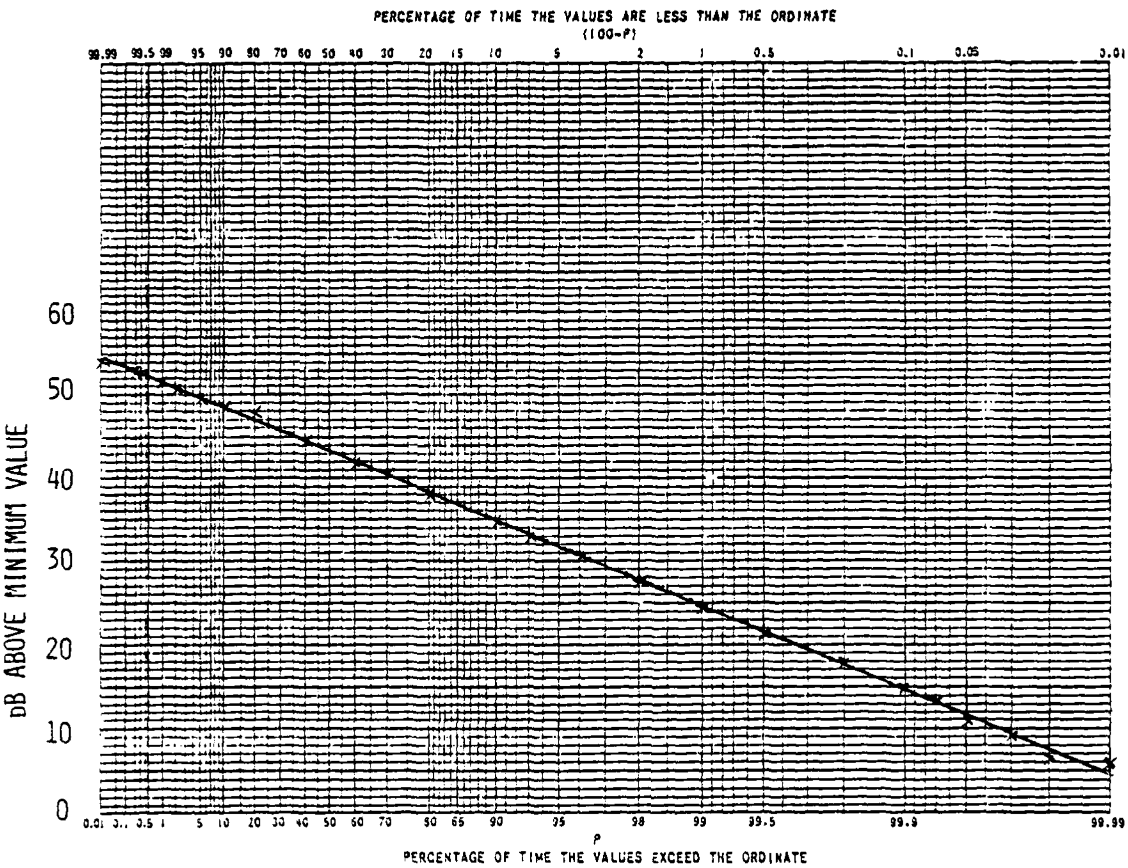

Fig. 2 — Cumulative amplitude distribution of fading path, generated by sum of 20 sinusoids with spread of ± 1 Hz

Table 4 — Example of random tone selection for spread of 1 Hz

NO.	FREQ. (Hz)	NO.	FREQ. (Hz)
1	-0.951	11	0.079
2	-0.878	12	0.169
3	-0.790	13	0.277
4	-0.638	14	0.381
5	-0.594	15	0.420
6	-0.404	16	0.599
7	-0.353	17	0.633
8	-0.261	18	0.763
9	-0.129	19	0.802
10	-0.048	20	0.994

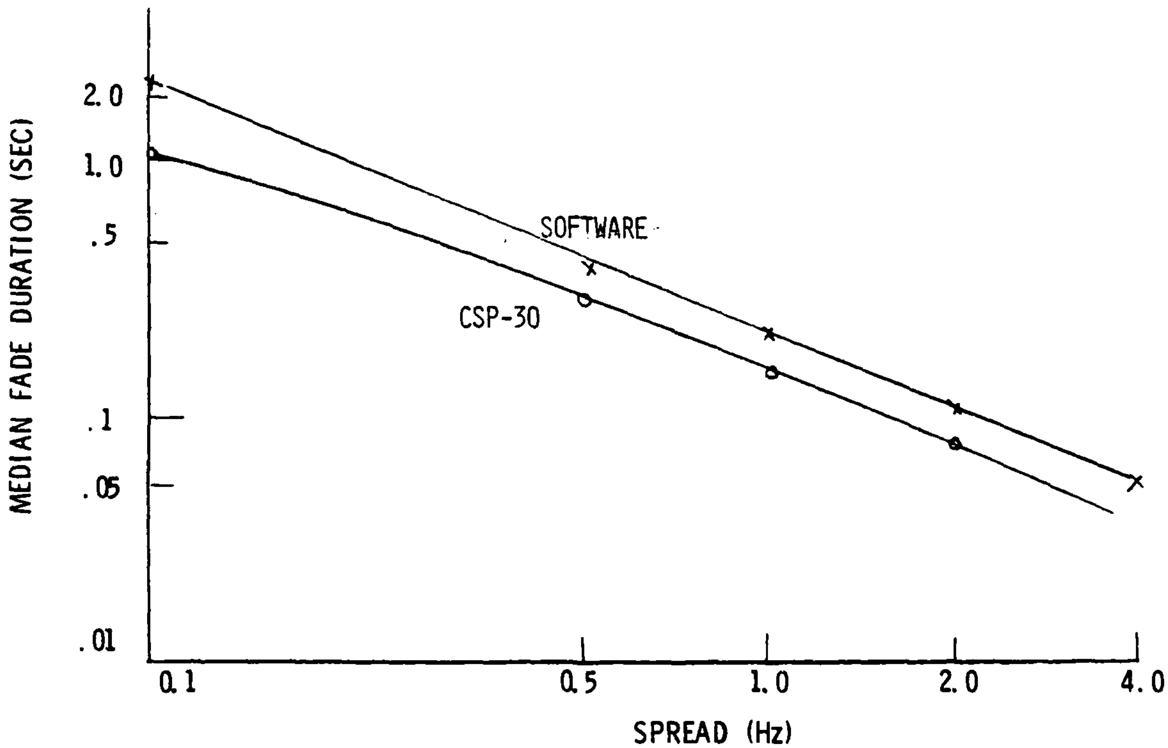

Fig. 3 — Median fade duration vs. frequency spread, one path, software channel and CSP-30

the level of the signal decreased to -6 dB relative to the mean, and the end of a fade was defined as when the level of the signal increased to -5 dB relative to the mean. Fade rate was defined as the average number of times the signal level left a fade condition in one minute. For the channel model described above, the average fade rate was 45 fades/minute measured over a 10 minute period. Figure 4 is a plot of fade rate versus spread for a one-path model combining 20 sinusoids at a modem sampling rate of 7200 Hz.

The simulation program provides for specifying up to ten paths. The path delays are specified in seconds, but they are interpreted in increments of the sampling period. The maximum amount of delay cannot exceed the period of the block of data processed by each pass through the subroutines. Thus, if data were processed in blocks of 100 samples at 7200 samples/second, the maximum delay would be 0.01389 seconds.

The table of variables contains the relative rms amplitude of each propagation path. The program calculates an amplitude scale factor for each propagation path which results in the received composite signal having the same rms level as the transmitted signal. The scale factor for each path is computed as

$$Gi = \frac{E_i}{\sqrt{\sum_{i=1}^{paths} E_i^2}}$$

where

E_i is the rms signal level for signal i

2.6 Doppler

In addition to the fixed doppler shift for each propagation path, the program provides for a variable doppler representing the composite of the frequency

translation errors and doppler shift due to the relative velocity between the transmitter and receiver platforms. This composite frequency error may be either fixed or time varying. Provisions are made for the selection of one of three types of varying doppler. They are:

- A step change in doppler of a specified amount to occur at a specified time after the start of the program. Time is computed in increments of the sampling period. This type of test represents the condition where there is an extremely rapid change in the relative velocity between the transmitter and receiver after a communication link has been established. It also may represent a step change in the frequency controls of the radio equipment. It provides a means to measure the ability of the modem receiver to accommodate a step change in frequency.
- A ramp change in doppler at a specified rate with the direction of the ramp reversing when high and low limits are reached. Each incremental change in doppler is equal to the rate of change divided by the sampling rate. The linearly changing doppler with time provides a means to test the doppler tracking capability of the modem receiver. Also, by changing the high and the low limits, it provides a means to test the effects of variations in the amplitude and phase delay due to the response of the receiver filter at the band edges.
- A high velocity fly-by scenario as diagramed in Figure 5. For this test the specifications include the velocity of

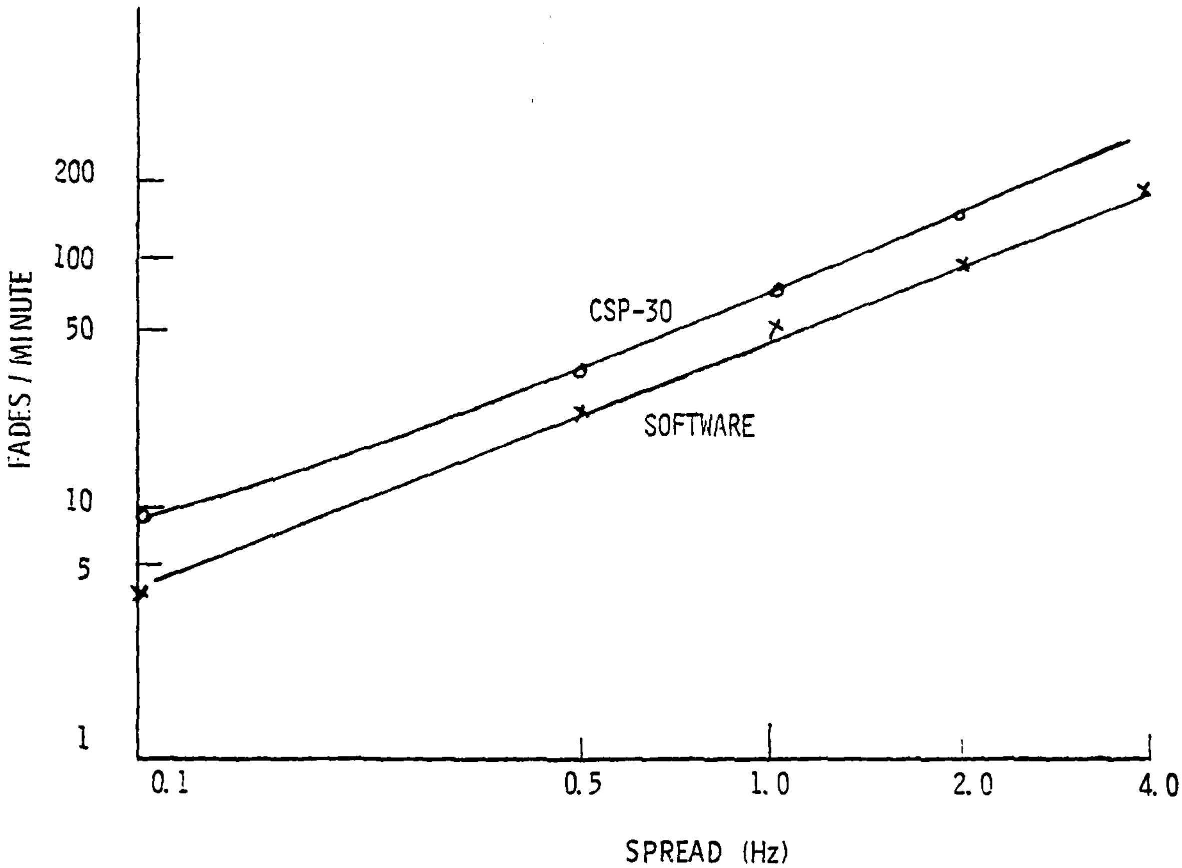

Fig. 4 — Fade rate vs. frequency spread, one path, software channel and CSP-30

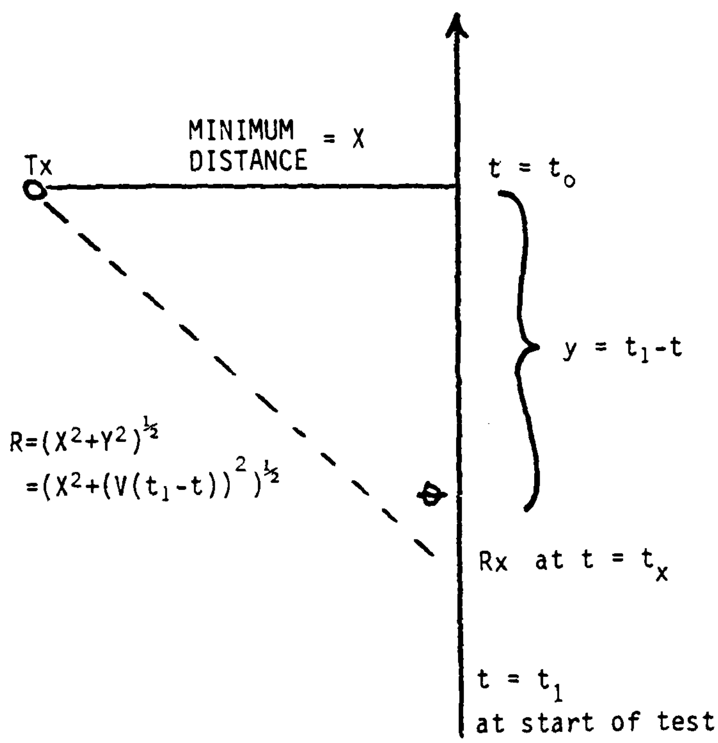

Fig. 5 — Fly-by scenario

the aircraft, the radio frequency, the minimum distance reached between transmitter and receiver at time t_o, the time (t_1) before t_o at which to start computing the doppler change, and that time relative to the start of the program (e.g., three seconds after the start of the program, initialize the fly-by test. At that time, the aircraft will be ten seconds from t_o. At t_o the distance between the transmitter and the receiver will be two miles. The aircraft velocity is 1000 miles per hour and the radio frequency is 10 MHz).

The general equation is:

$$\text{Doppler} = \frac{VF}{C} \cos(\theta)$$

$$= \frac{VF}{C} \frac{y}{(x^2 + y^2)^{\frac{1}{2}}}$$

$$= \frac{V^2F}{C} \frac{(t_1 - t)}{(x^2 + (V(t_1 - t))^2)^{\frac{1}{2}}}$$

With the aircraft velocity expressed in miles/hour, the radio frequency in MHz, the distance in miles, and time in seconds, the doppler may be calculated as follows:

$$\text{Doppler} = \left(\frac{V}{3600}\right)^2 \frac{F}{0.186} \frac{(t_1 - t)}{\left(x^2 + \left(\frac{V(t - t)}{3600}\right)^2\right)^{\frac{1}{2}}}$$

where:

Doppler is in Hertz

V is velocity in mph

F is frequency in MHz

t_1 is time away from t_o at start of test, in seconds

t is elapsed time in test, in seconds

X is distance between transmitter and receiver at time t_0, in miles.

Table 5 is a tabulation of the resulting doppler at increments of 10 seconds for the fly-by conditions given in the previous example. For this scenario, the maximum stress on the modem doppler tracking capability is at time t_0. Figure 6 shows the change in doppler at time t_0 versus aircraft velocity and distance between transmitter and receiver for a propagation frequency of 10 MHz.

2.7 Tonal Interference

Interference corresponding to other users of the radio channel is provided for by specifying tonal interference to be either CW, binary FSK, four-phase DPSK, or a saw-tooth swept audio tone. The number of tones, their frequency assignments, modulation baud periods, and rms power level are specified. If the tones are modulated, the information is random data, but the baud period must correspond to the same period as the block of data being processed.

The interference may be any combination of the four types. If two different types of modulated interference are specified, the conditions are limited to that in which they are both simultaneously keyed at an identical rate.

The interference is added to the signal after summation of the signals from each propagation path, and after frequency shifting all components. Thus, the interference is neither distorted by multipath nor is it frequency shifted. This represents the condition where there is always a stationary path between the interfering transmitter and modem receiver. For the case of a high velocity fly-by scenario with tonal interference, it represents transmission of the modem signal from the aircraft to the ground station, because receiving by the aircraft would require a doppler shift of the interference. Likewise, if there is a

Table 5 — Doppler incurred in fly-by example, Vel = 1000 mph, RF = 10 MHz, minimum distance = 2 miles

TIME BEFORE T_o (sec)	DOPPLER (Hz)	DOPPLER CHANGE (Hz/sec)
10	7.36	---
9	7.25	-0.11
8	7.11	-0.14
7	6.92	-0.19
6	6.65	-0.27
5	6.27	-0.38
4	5.71	-0.56
3	4.88	-0.83
2	3.67	-1.21
1	2.01	-1.66
0	0.00	-2.01

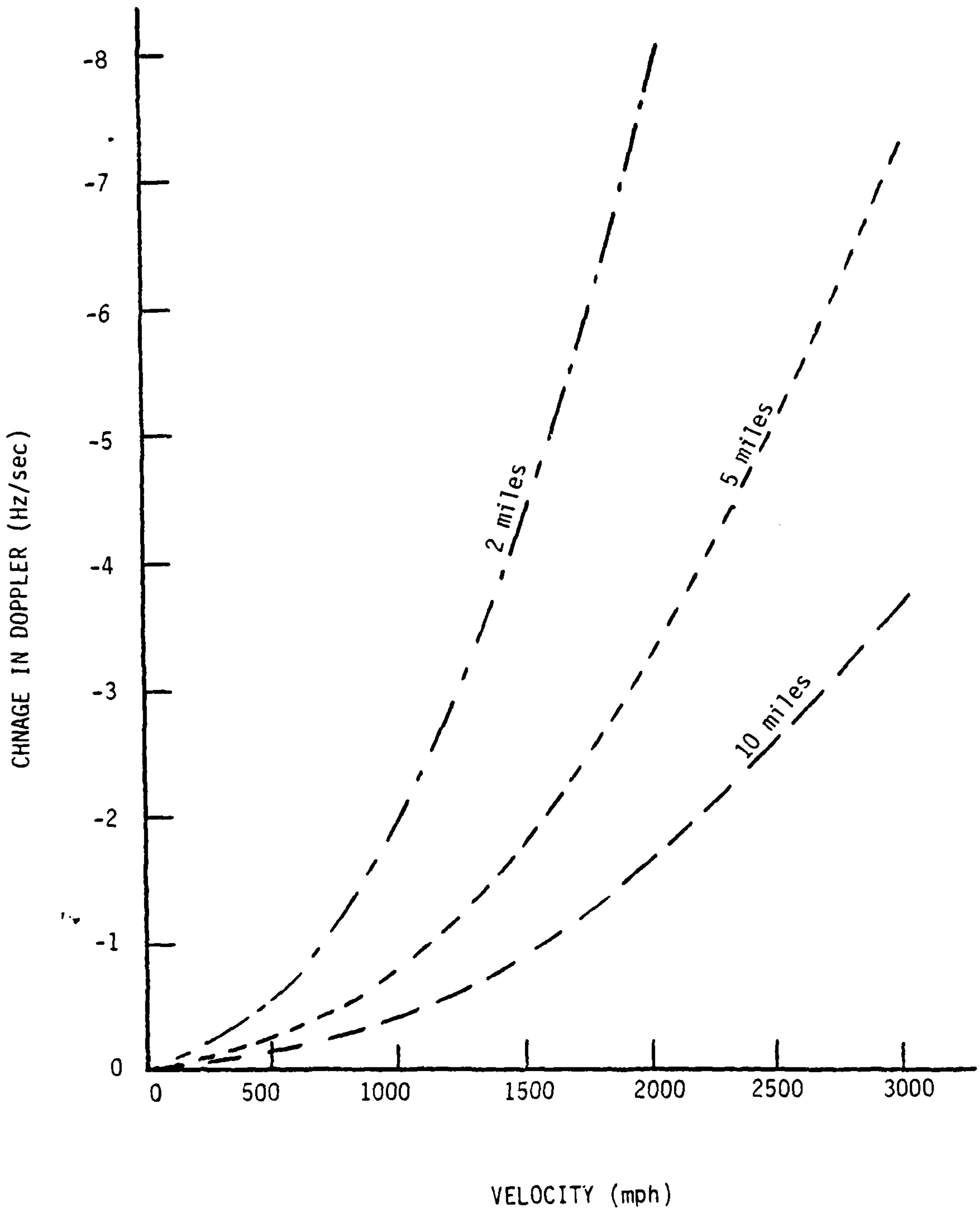

Fig. 6 — Doppler change during last second during fly-by scenario, as a function of velocity and distance, for a transmission frequency of 10 MHz

fixed frequency translation error plus tonal interference, the translation error must be due to the signal transmitter. If it is desired to represent a condition where the interfering transmitter has a frequency translation error, this may be done by making a corresponding change in the frequency assignments for the interference.

The level of the tonal interference is specified in dBm across 600 ohms. The program calculates the root mean square (RMS) voltage of the composite signal as follows:

as, $V(dBm) = 20 \log (V(RMS)/\sqrt{0.6})$

then, $V(RMS) = \sqrt{0.6}\ 10^{V(dBm)/20}$

This is normalized to the RMS voltage for zero dBm so,

$$V'(RMS) = V(RMS) / \sqrt{0.6}$$
$$= 10^{V(dBm)/20}$$

The peak amplitude of each interference tone is related to the RMS amplitude of the composite signal as follows:

$$E(peak) = V'(RMS)/\sqrt{N/2}$$
$$= \sqrt{2/N}\ 10^{V(dBm)/20}$$

The four types of tonal interference may occur in any combination as specified by an entry in the table of variables. The program converts that value from a floating point number to an integer with values of zero through 15. Zero indicates no interference is to be generated. The four least significant bits of that integer control the types of interference to be generated as follows:

Bit No. Set to 1	Interference Generated
0	CW
1	FSK
2	DPSK
3	Swept Tone

Table 6 lists the control word for each of the 15 possible combinations of tonal interference.

2.8 Pulse Interference

Provisions are made in the simulation program for a subroutine for the addition of non-Gaussian noise. At the present time an acceptable routine has not been developed. A dummy subroutine (Subroutine Pulse) is included with provisions for characterizing the noise pulses in terms of their RMS power level, duration, and the probability of occurence.

2.9 Gaussian Noise

The density of the continuous additive Gaussian noise is specified in dBm/Hz. The program converts that value to a noise scale factor which is used to multiply each Gaussian noise sample. Each noise sample is generated from a pair of random numbers, with uniform distribution between zero and one. The equation is:

$$X = ANZ\ (-2.0\ \ln(U))\ \cos\ (2\pi V)$$

where:

X = a sample of Gaussian noise

ANZ = noise scale factor

U and V = two uniformly distributed random numbers (zero to one).

With a noise scale factor of 1.0, the algorithm generates Gaussian noise with a long-term RMS value of 1.0 (zero mean and variance = 1). The scale factor required to generate the specified noise density level is calculated as follows:

$$P(\mathrm{dBm/Hz}) = 20 \log (ANZ) - 10 \log(SAMP/2)$$

where: P(dBm/Hz) is the specified noise density

ANZ is the noise scale factor

SAMP is the modem sampling rate

so, $$ANZ = 10^{(P(\mathrm{dBm/Hz}) + 10 \log (SAMP/2))/20}$$

Table 6 — Control words for selection of tonal interference

CONTROL WORD (OCTAL)	TYPES OF INTERFERENCE
00	None
01	CW
02	FSK
03	CW, FSK
04	DPSK
05	CW, DPSK
06	FSK, DPSK
07	CW, FSK, DPSK
10	SWEPT TONE
11	CW, SWEPT TONE
12	FSK, SWEPT TONE
13	CW, FSK, SWEPT TONE
14	DPSK, SWEPT TONE
15	CW, DPSK, SWEPT TONE
16	FSK, DPSK, SWEPT TONE
17	CW, FSK, DPSK, SWEPT TONE

2.10 AGC

The radio receiver automatic gain control characteristics may have a significant effect on the performance of some modems. Provisions are made in the simulation program for an AGC function, but at the present time this is just a dummy subroutine. The table of variables include the following AGC variables:

Var (2,2) - current AGC gain value

Var (5,2) - constant controlling AGC attack time

Var (6,2) - constant controlling AGC decay time

Var (7,2) - constant representing maximum AGC amplification

2.11 Input Variables Modified by Program

Each test is controlled by the variables read from a disk file into the two dimensional array VAR (10, 10). These values are all entered as floating point numbers. The following is a list of those variables that are converted to integer by the program, but otherwise remain unchanged during the entire test:

1. VAR (2, 1) = NBD
2. VAR (3, 1) = NPATH
3. VAR (4, 1) = NFADE
4. VAR (5, 1) = KDOPP
5. VAR (6, 1) = KINTF
6. VAR (7, 1) = KPULSE
7. VAR (8, 1) = KTXFIL
8. VAR (9, 1) = KRXFIL
9. VAR (10, 1) = KAGC
10. VAR (1, 5) = NCW
11. VAR (1, 6) = NFSK
12. VAR (1, 7) = NPSK
13. VAR (3, 2) = NTAPE
14. VAR (4, 2) = MTAPE

The interference levels of the tonal interference are specified in dBm. The program converts these to the peak amplitude of each tone. The variables involved are:

1. VAR (4, 5) = AMPCW
2. VAR (5, 6) = AMPFSK
3. VAR (4, 7) = AMPPSK
4. VAR (4, 8) = AMPSAW

The interference levels for the Gaussian noise and the noise pulse are specified in dBm/HZ. The program converts these values to scale factors. The variables involved are:

1. VAR (1, 2) = GAUSS
2. VAR (2, 10) = AMPPUL

The path delays are specified in seconds. The program multiplies these by the sampling rate to get the delay in terms of the number of sample periods. The variables involved are in VAR(I, 3), where I refers to the path number.

The propagation paths are defined by their relative RMS amplitudes. The program converts these to scale factors dependent on the number of paths. The variables affected are in VAR (I, 4), where I refers to the path number.

3.0 Conclusions

A FORTRAN program has been developed to provide a simple, standardized means of applying uniform tests to various hf modem signals. It has been applied to testing of the ANDVT hf modem signal design, the results of which are published in reference [16]. The program provides a beneficial supplement to real-time testing and to Monte-Carlo type non-real-time tests.

4.0 References

1. Voelcker, H. B., "Phase-Shift Keying in Fading Channels", Proc. IEE (British), Vol. 107, Part B, January 1960, pp. 31-38.

2. Bello, P. A., "Error Probabilities Due to Atmospheric Noise and Flat Fading in HF Ionospheric Communication Systems", IEEE Trans. Comm. Technology, Vol. COM-13, September 1965, pp. 266-279.

3. Proakis, J. G., "Probabilities of Error for Adaptive Reception of M-Phase Signals", IEEE Trans. Commun. Tech. COM-16, pp. 71-81.

4. Bello, P. A., "Characterization of Randomly Time-Varient Linear Channels", IEEE Trans. Comm. Systems, CS-11, pp. 360-393, 1963.

5. Richman, S. H. and P. Monsen, "HF Channel Effects in Performance of Carlos II", Final Report, Project RI-72-0002, Signatron, Inc., Lexington, MA, 1972.

6. Fruedberg, R., "A Laboratory Simulator for Frequency-Selective Fading", IEEE International Communication Conference, Boulder, CO, 7-9 June 1965.

7. Walker, W. F., "A Sample Baseband Fading Multipath Channel Simulator", IEEE International Communication Convention, Boulder, CO, 7-9 June 1965.

8. Watterson, C. C., et al., "An Ionospheric Channel Simulator", ESSA Tech. Memo, ERLTM-ITS 198.

9. Watterson, C. C., et al., "Experimental Confirmation of an HF Channel Model", IEEE Trans. Comm. Tech. COM-18, pp. 792-803, December

10. Watterson, C. C. and Minister, C. M., "HF Channel-Simulator Measurements and Performance Analyses on the USC-10, ACQ-6, and MX-190 PSK Modems", OT Report 75-56, pp. 1-265.

11. Chapin, E. W. and Roberts, W. K. "A Radio Propagation and Fading Simulator Using Radio Frequency Acoustic Waves in a Liquid", Proc. IEEE, Vol. 54, 6, 1072.

12. Clarke, K. K., "Random Channel Simulation and Instrumentation", IEEE First Annual Communication Convention, Boulder, CO, pp. 623-629.

13. Kahler, F. C., "A High-Frequency-Radio Ionospheric Propagation-Path Simulator", NRL Memorandum Report 1969, April 1969.

14. Goldberg, B., et al., "Stored Ionosphere", IEEE First International Communication Convention, Boulder, CO, 7-9 June 1965.

15. Pinto, R. W. and Bello, P. A., "HF Channel Simulation", CNR Report No. 14, Contract N00014-74-C-0048, CNR, Inc., Newton, MA, 1974.

16. Wayne Jewett and Raymond Cole, Jr., "Non-Real Time Stress Tests of the ANDVT HF Modem", NRL Memorandum Report in Publication.

5.0 List of Illustrations

Tables

Figures

```
C                       APPENDIX A
C                       PROGRAM STRESS.FTN
C       TO ADD STRESS TO MODEM SIGNAL
C       IN FORM OF GAUSSIAN NOISE, DOPPLER, FILTERING
C       NARROWBAND INTERFERENCE, PULSE, MULTIPATH
C
C       INPUT IS DIGITIZED MODULATOR OUTPUT ON MT: OR DISK
C       FILE TAN.DAT
C       OUTPUT IS DIGITIZED INPUT TO DEMODULATOR
C       ON MT: OR DISK FILE STRESS.DAT
C
C       REQUIRED DATA FILES FROM DISK ARE:
C               STRVAR.DAT TABLE OF VARIABLES
C               PATH.DAT RAYLIEGH PATH DATA
C
        COMMON/BLK1/VAR(10,10),A(100),Q(100)
        COMMON/BLK2/TXFIL(200)
        COMMON/BLK3/DELAY(2,200)
        COMMON/BLK4/PHASE(40),TAU(4),FQSAW
        COMMON/BLK5/PATH(1024),WT,TIMEX
        COMMON/BLK6/I1,I2
        COMMON/BLK7/FLAG,KFLAG
        COMMON/BLK8/IA(1000),ISTAT,NDX,BLOCK
        COMMON/BLK9/SAVIN(3,2),SAVOUT(3,2)
        COMMON/BLK10/RXFIL(200)
        COMMON/BLK11/AGCFIL(2,100)
        COMMON/BLK12/IB(1000),JSTAT,NDZ,KCODE,RMS,PEAK,TOTAL
        COMMON/BLK13/DELTA,ROLD(10),RNEW(10),GOLD(10),GNEW(10),NDY,KSTAT
        COMPLEX PATH
        EQUIVALENCE  (VAR(1,1),SAMP),(VAR(2,10),AMPPUL),(VAR(1,2),GAUSS)
        EQUIVALENCE (VAR(4,5),AMPCW),(VAR(5,6),AMPFSK),(VAR(4,7),AMPPSK)
        EQUIVALENCE (VAR(1,8),F1SAW),(VAR(4,8),AMPSAW)
C
        WRITE(5,1)
    1   FORMAT(1X,'ENTER NO. OF FRAMES',/)
        READ(5,2)FRAMES
    2   FORMAT(F8.0)
        WRITE(5,3)
    3   FORMAT(1X,'ENTER NO. OF BLOCKS TO SKIP, STARTING POINT',/)
        READ(5,4)NSK
        READ(5,4)NDX
    4   FORMAT(I5)
        NSK=NSK+1
C                       READ TABLE OF VARIABLES FROM DISK
        CALL ASSIGN(4,'STRVAR.DAT')
        READ(4,END=3000,ERR=3000)VAR
        CALL CLOSE (4)
C
C                       SET INITIAL CONDITIONS
        NBD=VAR(2,1)
        NPATH=VAR(3,1)
        NFADE=VAR(4,1)
```

```
      KDOPP=VAR(5,1)
      KINTF=VAR(6,1)
      KPULSE=VAR(7,1)
      KTXFIL=VAR(8,1)
      KRXFIL=VAR(9,1)
      KAGC=VAR(10,1)
      NCW=VAR(1,5)
      NFSK=VAR(1,6)
      NPSK=VAR(1,7)
      NTAPE=VAR(3,2)
      MTAPE=VAR(4,2)
      DO 5 I=1,4
    5 TAU(I)=0.
      KFLAG=0
      RMS=0.
      PEAK=0.
      TOTAL=0.
      FLAG=0.
      I1=0
      I2=0
      FRAME=0.
      NDZ=0
      FQSAW=F1SAW
      WT=0.
      KCODE=3
      TIMEX=0.
C                         RANDOM PHASES FOR DPSK TONES
      DO 7 I=1,40
      X=RAN(I1,I2)
    7 PHASE(I)=2.0*(X-0.5)*3.14159
C                                 ASSIGN AND SET INPUT DEVICE
C                                 NTAPE=4 IS NO INPUT
      IF(NTAPE.EQ.4)GO TO 15
      IF(NTAPE.EQ.3)GO TO 10
C                                 INPUT FROM MT:
      CALL ASSIGN(3,'MT:')
      DO 9 I=1,NSK
      ISTAT=0
      CALL MTIN(3,NTAPE,1000,IA,ISTAT)
      IF(ISTAT.EQ.1)GO TO 3000
    9 CONTINUE
      GO TO 15
C                                 INPUT FROM DISC FILE
   10 CALL ASSIGN(3,'TAN.DAT')
      DO 11 I=1,NSK
      READ(3,END=3000,ERR=3000)IA
   11 CONTINUE
C                                 ASSIGN OUTPUT DEVICE
C                                 SET MTOUT FOR D/A
   15 IF(MTAPE.LE.1)CALL ASSIGN(2,'MT:')
      IF(MTAPE.EQ.3)CALL ASSIGN(2,'STRESS.DAT')
      IF(MTAPE.LE.1)CALL MTSET(2,MTAPE,1,ISTAT)
      BLOCK=0.
C
C                                 CALCULATE PEAK AMPL OF CW TONES
```

```
      IF(NCW.LE.0)GO TO 16
      AMPCW=SQRT(2./NCW)*10.**(AMPCW/20.)
C                                CALCULATE PEAK AMPL OF FSK TONES
   16 IF(NFSK.LE.0)GO TO 17
      AMPFSK=SQRT(2./NFSK)*10.**(AMPFSK/20.)
C                                CALCULATE PEAK AMPL OF PSK TONES
   17 IF(NPSK.LE.0)GO TO 18
      AMPPSK=SQRT(2./NPSK)*10.**(AMPPSK/20.)
C                                CALCULATE PEAK AMPL OF SWEPT TONE
   18 AMPSAW=SQRT(2.)*10.**(AMPSAW/20.)
C                                CALCULATE GAUSSIAN NOISE SCALE FACTOR
      Z=10.*ALOG10(SAMP/2.)
      GAUSS=10.**((GAUSS+Z)/20.)
C                                CALCULATE NOISE PULSE SCALE FACTOR
      AMPPUL=10.**((AMPPUL+Z)/20.)
C                                CALCULATE AMPL SCALE FACTOR FOR
C                                        EACH DELAY PATH
C                                CONVERT DELAY IN SEC. TO NO. OF SAMPLES
      IF(NPATH.EQ.1)GO TO 25
      R=0.
      DO 22 I=1,NPATH
   22 R=R+VAR(I,4)*VAR(I,4)
      IF(R.LT.0.001)R=0.001
      R=SQRT(R)
      DO 23 I=1,NPATH
   23 VAR(I,4)=VAR(I,4)/R
   25 CONTINUE
      DO 29 I=1,NPATH
   29 VAR(I,3)=VAR(I,3)*SAMP
C                                IF FADING CHANNEL READ IN PATH DATA
C                                PATH DATA GENERATED AT 0.01*SAMP
C                                STORE OLD AND NEW VALUES
      IF(NFADE.EQ.0)GO TO 28
      CALL ASSIGN(4,'PATH.DAT')
      READ(4,END=3000,ERR=3000)PATH
      NDY=0
      DELTA=0.
      DO 27 J=1,2
      DO 26 I=1,NPATH
      NDY=NDY+1
      ROLD(I)=RNEW(I)
      GOLD(I)=GNEW(I)
      RNEW(I)=REAL(PATH(NDY))
      GNEW(I)=AIMAG(PATH(NDY))
   26 CONTINUE
   27 CONTINUE
   28 CONTINUE
C
C-------------------------------------
C
C                         MAIN LOOP
   20 CONTINUE
      FRAME=FRAME+1.0
C                                IF NO INPUT ,BYPASS ROUTINES
C                                WHICH PROCESS INPUT DATA
```

```
      IF(NTAPE.LT.4)GO TO 40
      DO 30 I=1,NBD
   30 A(I)=0.
      GO TO 50
   40 CONTINUE
C                                READ INPUT DATA
      ISTAT=0
      CALL INPUTX
      IF(ISTAT.EQ.1)GO TO 3000
C                                TRANSMITTER FILTER
      CALL TXFILT
C                                ALL PASS FILTER
      CALL ALPASX
      KSTAT=0
C                                COMBINE OUTPUTS FROM N PATHS
      CALL DELAYX
      IF(KSTAT.EQ.1)GO TO 3000
C                                FREQ TRANSLATE COMPOSITE SIGNAL
C                                DOPPLER FREQ CHANGES DUE TO
C                                SCENARIO MADE IN SUBROUTINE DOPVAR
C                                WHICH IS CALLED FROM FREQX
      CALL FREQX
   50 CONTINUE
C                                ADD NARROWBAND TONAL INTERFERENCE
      CALL INTERF
C                                ADD NOISE PULSE
      CALL PULSE
C                                ADD GAUSSIAN NOISE
      CALL NOISEX
C                                RECEIVER FILTER
      CALL RXFILT
C                                RECEIVER GAIN CONTROL
      CALL AGC
C                       STORE OUTPUT
      CALL STOREX
C                                TEST FRAME COUNT
      IF(FRAME.LT.FRAMES)GO TO 20
C
C------------------------------------
C                                LOOP COMPLETED
 3000 CONTINUE
      WRITE(5,3001)TOTAL
 3001 FORMAT(1X,'TOTAL NO. OF SAMPLES=',F9.0)
      IF(TOTAL.LE.1.)GO TO 3100
      RMS=SQRT(RMS/TOTAL)
      V=RMS*5./2047.
      DBM=10.*ALOG10(V*V/0.6)
      DB=10.*ALOG10(PEAK/RMS)
      WRITE(5,3002)PEAK,DB,DBM
 3002 FORMAT(1X,'PEAK=',F9.2,' PEAK/RMS(DB)=',F9.2,' DBM=',F8.3)
 3100 CONTINUE
      IF(MTAPE.GT.1)GO TO 3200
      CALL MTOUT(2,MTAPE,1000,IB,ISTAT)
      CALL MTSET(2,MTAPE,0,ISTAT)
      CALL MTOUT(2,MTAPE,0,IB,ISTAT)
```

```
      CALL MTOUT(2,MTAPE,-1,IB,ISTAT)
      GO TO 3300
 3200 WRITE(2)IB
      ENDFILE 2
 3300 CONTINUE
      IF(NTAPE.LE.1)CALL MTIN(3,NTAPE,-1,IA,ISTAT)
      IF(NTAPE.EQ.3)CALL CLOSE(3)
      IF(NFADE.GT.0)CALL CLOSE(4)
      END
C     INPUT ONE FRAME OF ANALOG SAMPLES
C
      SUBROUTINE INPUTX
      COMMON/BLK1/VAR(10,10),A(100),Q(100)
      COMMON/BLK8/IA(1000),ISTAT,NDX,BLOCK
C
C                                 READ IN NBD SAMPLES
C                                 CONVERT TO FLOATING POINT
C                                 NORMALIZE TO ZERO DBM(V=0.77459667)
C                                 E=V*2047/5 =317.12
      NBD=VAR(2,1)
      NTAPE=VAR(3,2)
      DO 10 I=1,NBD
      NDX=NDX+1
      A(I)=FLOAT(IA(NDX)/16)/317.12
      IF(NDX.LT.1000)GO TO 10
      NDX=0
      ISTAT=0
      IF(NTAPE.LE.1)CALL MTIN(3,NTAPE,1000,IA,ISTAT)
C                       CHECK FOR SUCCESSFUL READ
      IF(ISTAT.EQ.0)GOTO 4
C                       CHECK FOR EOF ON TAPE
      IF(ISTAT.EQ.1)GOTO 30
C                       CHECK FOR END OF TAPE
      IF(ISTAT.NE.2)GOTO 3
      WRITE(5,22)
22    FORMAT(' END OF TAPE')
      GO TO 30
C                       TAPE READ ERROR
3     WRITE(5,33)ISTAT
33    FORMAT(' TAPE READ ERROR   ISTAT=:',I5)
4     IF(NTAPE.EQ.3)READ(3,END=30,ERR=30)IA
      BLOCK=BLOCK+1.0
C                       CHECK BLOCKER CODE
      K=3
      DO 8 J=1,1000
      KB=IA(J).AND.15
      IF(K.EQ.KB)GO TO 7
      WRITE(5,5)BLOCK
    5 FORMAT(1X,'BLOCKER CODE ERROR IN BLOCK',F8.0)
      GO TO 10
    7 K=K+4
      IF(K.GT.15)K=3
    8 CONTINUE
   10 CONTINUE
      RETURN
```

```
30    WRITE(5,35) BLOCK
35    FORMAT(1X,'EOF OR ERROR ON ANALOG INPUT IN BLOCK',F8.0,/)
      ISTAT=1
      RETURN
      END
      SUBROUTINE MTIN(LUN,IDRIVE,ICNT,IARRAY,ISTAT)
C     MTIN MAG-TAPE DRIVER
C     D. TATE   ORI,INC   JAN 79
C     THIS SUBROUTINE IS USED TO READ FROM MAG TAPE
C     (NON-ANSI). VARIABLES ARE DEFINED IN I/O DRIVERS
C     REFERENCE MANUAL, SUBROUTINES ARE DEFINED IN
C     EXECUTIVE REFERENCE MANUAL.
C     PARAMETERS:
C     LUN = LOGICAL UNIT NUMBER (POSITION IN LOGICAL
C             UNIT TABLE)
C     IDRIVE = DRIVE NUMBER (0 OR 1)
C     ICNT < 0 => REWIND REQUEST
C          > 0 => NUMBER IF INTEGER WORDS IN RECORD
C     IARRAY = ARRAY IN WHICH DATA FROM TAPE WILL BE STORED
C     ISTAT = STATUS RETURNED TO CALLING PROGRAM
C           = 0 => O.K.
C           = 1 => END OF FILE DETECTED
C           = 2 => END OF TAPE DETECTED
C           = 3 => DATA OVERRUN (BUFFER SIZE TOO SMALL)
C           < 0 => UNSPECIFIED QIO$ ERROR
C
      DIMENSION ISB(2),IPRL(6),IARRAY(1)
      DATA ISSUC/1/,IOATT/"1400/,IEDAA/-8/IORWD/"2400/
      DATA IORLB/"1000/,IEDAO/-13/,IEEOF/-10/,IEEOT/-62/
C ASSIGN LUN TO MAG TAPE UNIT
      CALL ASNLUN(LUN,'MT',IDRIVE,ISTAT)
      IF(ISTAT.NE.ISSUC) GOTO 99
C CLEAR PARAMETER LIST
      DO 10 I=1,6
10    IPRL(I)=0
C REQUEST EXCLUSIVE USE OF TAPE DRIVE
      CALL QIO(IOATT,LUN,1,0,ISB,IPRL,ISTAT)
C SEE IF QIO DIRECTIVE ACCEPTED
      IF(ISTAT.NE.ISSUC) GOTO 99
C QIO DIRECTIVE ACCEPTED, NOW WAIT FOR I/O TO FINISH
      CALL WAITFR(1,ISTAT)
C CHECK WAITFR DIRECTIVE ACCEPTANCE
      IF(ISTAT.NE.ISSUC) GOTO 99
C DIRECTIVE ACCEPTED, NOW CHECK I/O STATUS
      IF(ISB(1).EQ.ISSUC) GOTO 20
C IF DEVICE ALREADY ATTACHED, CONTINUE
C SIGN EXTEND FOR ERROR CODES
      ISTS=ISB(1)-256
      IF(ISTS.NE.IEDAA) GOTO 99
C CHECK ICNT FOR FUNCTION
20    IF(ICNT.GE.0) GOTO 30
C REWIND REQUEST
      CALL QIO(IORWD,LUN,2,0,ISB,IPRL,ISTAT)
      GOTO 80
C NORMAL READ RECORD REQUEST
```

```
30      CALL GETADR(IPRL(1),IARRAY(1))
        IPRL(2)=ICNT*2
        CALL QIO(IORLB,LUN,2,0,ISB,IPRL,ISTAT)
C CHECK DIRECTIVE ACCEPTANCE FOR ALL FUNCTIONS
80      IF(ISTAT.NE.ISSUC) GOTO 99
        CALL WAITFR(2,ISTAT)
        IF(ISTAT.NE.ISSUC) GOTO 99
C FUNCTION WAS ACCEPTTED, NOW CHECK FINAL STATUS
        ISTAT=256
        IF(ISB(1).EQ.ISSUC) ICNT=ISB(2)/2
        IF(ISB(1).EQ.ISSUC) GOTO 99
C SIGN EXTEND STATUS FOR ERROR CODES
        ISTS=ISB(1)-256
        IF(ISTS.EQ.IEEOF) ISTAT=257
        IF(ISTS.EQ.IEEOT) ISTAT=258
        IF(ISTS.EQ.IEDAO) ISTAT=259
C NOT EOF, EOT OR DAO, BUT STILL AN ERROR
        IF(ISTAT.EQ.256) ISTAT=ISB(1)
99      ISTAT=ISTAT-256
        RETURN
        END
C       ALLPASS FILTER
C
        SUBROUTINE ALPASX
C                                    ALL PASS FILTER
C   OUTPUT OF PHASE SHIFTER 1 IS IN ARRAY Q
C      "     "     "       "    2  "   "    "    A
        COMMON/BLK1/VAR(10,10),A(100),Q(100)
        COMMON/BLK9/SAVIN(3,2),SAVOUT(3,2)
        DIMENSION X1(101,2),X2(100),HC(3,2)
        DATA HC/-.6074647,.3229095,.8083183,.9418089,.6132104,-.0730881/
        NBD=VAR(2,1)
        DO 15 I=1,NBD
   15   X2(I)=A(I)
C  FILTER NBD POINTS
C  LA=1, Q PHASE
C  LA=2, I PHASE
        NBD1=NBD+1
        DO 10 LA=1,2
        DO 12 I=1,NBD
   12   X1(I+1,1)=X2(I)
        KIN=2
        DO 30 ISEC=1,3
        KOUT=KIN
        KIN=KIN+1
        IF(KIN.GT.2)KIN=1
        X1(1,KIN)=SAVIN(ISEC,LA)
        X1(1,KOUT)=SAVOUT(ISEC,LA)
        DO 20 I=2,NBD1
   20   X1(I,KOUT)=HC(ISEC,LA)*(X1(I,KIN)+X1(I-1,KOUT))-X1(I-1,KIN)
        SAVIN(ISEC,LA)=X1(NBD1,KIN)
   30   SAVOUT(ISEC,LA)=X1(NBD1,KOUT)
        GO TO (40,50),LA
   40   DO 45 I=1,NBD
   45   Q(I)=X1(I+1,KOUT)
```

```
   10   CONTINUE
   50   DO 200 I=1,NBD
  200   A(I)=X1(I+1,KOUT)
 3000   RETURN
        END
C       ADD GAUSSIAN NOISE TO ANALOG SAMPLES
C
        SUBROUTINE NOISEX
        COMMON/BLK1/VAR(10,10),A(100),Q(100)
        COMMON/BLK6/I1,I2
        EQUIVALENCE (VAR(1,2),GAUSS)
C
        NBD=VAR(2,1)
        DO 10 I=1,NBD
        U=RAN(I1,I2)
        V=RAN(I1,I2)
        U=GAUSS*SQRT(2.0*ABS(ALOG(U)))
        V=6.283185308*V
        A(I)=A(I)+U*COS(V)
   10   CONTINUE
        RETURN
        END
C       FREQUENCY TRANSLATION BY DOPP
C
        SUBROUTINE FREQX
        COMMON/BLK1/VAR(10,10),A(100),Q(100)
        COMMON/BLK5/PATH(1024),WT,TIMEX
        EQUIVALENCE (VAR(1,1),SAMP),(VAR(10,9),DOPP)
        COMPLEX PATH
        NBD=VAR(2,1)
        TWOPI=6.283185308
        TS=TWOPI/SAMP
        DO 50 I=1,NBD
C                                   UPDATE VARIATIONS IN DOPPLER
        CALL DOPVAR
        WT=WT+DOPP*TS
        IF(WT.GT.TWOPI)WT=WT-TWOPI
        IF(WT.LT.-TWOPI)WT=WT+TWOPI
   50   A(I)=A(I)*COS(WT)+Q(I)*SIN(WT)
        RETURN
        END
C                TAP DELAY LINE
        SUBROUTINE DELAYX
        COMMON/BLK1/VAR(10,10),A(100),Q(100)
        COMMON/BLK3/DELAY(2,200)
        COMMON/BLK5/PATH(1024),WT,TIMEX
        COMMON/BLK13/DELTA,ROLD(10),RNEW(10),GOLD(10),GNEW(10),NDY,KSTAT
        COMPLEX PATH
        NFADE=VAR(4,1)
        IF(NFADE.EQ.0)RETURN
        NBD=VAR(2,1)
        NPATH=VAR(3,1)
C                           ADD DATA TO DELAY LINE, CLEAR INPUT
        DO 50 I=1,NBD
```

```
      DELAY(1,I+NBD)=A(I)
      DELAY(2,I+NBD)=Q(I)
      A(I)=0.
   50 Q(I)=0.
C                               ADD SAMPLES FROM EACH PATH
C                     RELATIVE AMPL OF EACH PATH IS IN VAR(J,4)
C                     PATH GAIN VALUES SAMPLED AT 0.01*SAMP
C                     RELATIVE DELAY OF EACH PATH IS IN VAR(J,3)
C                     DELAY GIVEN IN NO. OF SAMPLES
      KSTAT=0
      DO 150 I=1,NBD
      DELTA=DELTA+0.01
      DO 100 J=1,NPATH
      N=NBD-VAR(J,3)
C                     INTERPOLATE BETWEEN PATH VALUES
      R=ROLD(J)+(RNEW(J)-ROLD(J))*DELTA
      G=GOLD(J)+(GNEW(J)-GOLD(J))*DELTA
      X=DELAY(1,N+I)*R+DELAY(2,N+I)*G
      A(I)=A(I)+X*VAR(J,4)
      X=DELAY(2,N+I)*R-DELAY(1,N+I)*G
      Q(I)=Q(I)+X*VAR(J,4)
  100 CONTINUE
      IF(DELTA.LT.1.)GO TO 150
C                               READ IN NEW PATH VALUES
      DELTA=0.
      DO 110 J=1,NPATH
      NDY=NDY+1
      ROLD(J)=RNEW(J)
      GOLD(J)=GNEW(J)
      RNEW(J)=REAL(PATH(NDY))
      GNEW(J)=AIMAG(PATH(NDY))
      IF(NDY.LT.1024)GO TO 110
      NDY=0
      READ(4,END=3000,ERR=3000)PATH
  110 CONTINUE
  150 CONTINUE
C                               SHIFT SIGNAL IN DELAY LINE
      DO 200 I=1,NBD
      DELAY(1,I)=DELAY(1,I+NBD)
  200 DELAY(2,I)=DELAY(2,I+NBD)
      RETURN
 3000 WRITE(5,4000)
 4000 FORMAT(1X,'EOF ON PATH DATA',/)
      KSTAT=1
      RETURN
      END
C
C
C             UPDATE VARIATION IN DOPPLER
C             3 ALGOTHRIMS: STEP,RAMP,AND FLYBY
      SUBROUTINE DOPVAR
      COMMON/BLK1/VAR(10,10),A(100),Q(100)
      COMMON/BLK5/PATH(1024),WT,TIMEX
      COMMON/BLK7/FLAG,KFLAG
      EQUIVALENCE (VAR(1,1),SAMP),(VAR(1,9),START),(VAR(2,9),DOPX)
```

```
      EQUIVALENCE (VAR(3,9),RATE),(VAR(4,9),DOPLOW),(VAR(5,9),DOPHGH)
      EQUIVALENCE (VAR(6,9),TIMEY),(VAR(7,9),XDIST),(VAR(8,9),VEL)
      EQUIVALENCE (VAR(9,9),RF),(VAR(10,9),DOPP)
      COMPLEX PATH
C
      KDOPP=VAR(5,1)
      IF(KDOPP.EQ.0)RETURN
      GO TO (1,2,3),KDOPP
C                    STEP CHANGE IN DOPPLER AT TIME "START"
C                    TEST IF STEP HAS OCCURRED
    1 IF(KFLAG.EQ.1)RETURN
      TIMEX=TIMEX+1.0/SAMP
      IF(TIMEX.LT.START)RETURN
      DOPP=DOPP+DOPX
      KFLAG=1
      RETURN
C
C                    RAMP CHANGE IN DOPPLER AT CONSTANT RATE
C                    MAX LIMIT= DOPLOW,DOPHGH
    2 DOP=DOP+RATE/SAMP
      IF(DOPP.GE.DOPHGH)RATE=-1.0*RATE
      IF(DOPP.LE.DOPLOW)RATE=-1.0*RATE
      RETURN
C
C                    HIGH VELOCITY FLYBY
C                    BEGIN TEST AT TIME START
C                    TIMEY IS TIME PLANE IS FROM MIN DISTANCE
C                    XDIST IS MIN DISTANCE
C                    VEL IS MPH
C                    RF IS MHZ
    3 TIMEX=TIMEX+1.0/SAMP
      IF(TIMEX.LT.START)RETURN
      V=(VEL/3600.)**2
      X=XDIST*XDIST
      T=TIMEY*TIMEY
      X1=SQRT(X+V*T)
      DOPP=V*RF*TIMEY/X1/0.186
      TIMEY=TIMEY-1.0/SAMP
      RETURN
      END
C
C                                        ADD MULTITONE INTERFERENCE
C                            CW , FSK , 4-PHASE DPSK, SWEPT TONE
C                            MAX OF 40 TONES IN FSK AND DPSK MODES
      SUBROUTINE INTERF
      COMMON/BLK1/VAR(10,10),A(100),Q(100)
      COMMON/BLK4/PHASE(40),TAU(4),FQSAW
      COMMON/BLK6/I1,I2
      DIMENSION GRAY(4)
      EQUIVALENCE (VAR(1,1),SAMP),(VAR(2,5),F1CW),(VAR(3,5),FXCW)
      EQUIVALENCE (VAR(4,5),AMPCW),(VAR(2,6),F1MK),(VAR(3,6),SHIFT)
      EQUIVALENCE (VAR(4,6),FXFSK),(VAR(5,6),AMPFSK),(VAR(2,7),F1PSK)
      EQUIVALENCE (VAR(3,7),FXPSK),(VAR(4,7),AMPPSK),(VAR(1,8),F1SAW)
      EQUIVALENCE (VAR(2,8),F2SAW),(VAR(3,8),RSAW),(VAR(4,8),AMPSAW)
      DIMENSION FFSK(40)
```

```
      DATA GRAY/1.0,3.0,-1.0,-3.0/
C
      KINTF=VAR(6,1)
      IF(KINTF.EQ.0)RETURN
      PI=3.14159
      TWOPI=2.0*PI
      NBD=VAR(2,1)
      NCW=VAR(1,5)
      NFSK=VAR(1,6)
      NPSK=VAR(1,7)
      TS=1./SAMP
      M=KINTF.AND.1
      IF(M.EQ.0)GO TO 2
C                                   CW INTERFERENCE
C                             "NCW" TONES SPACED "FXCW" HZ APART
C                             PEAK AMPL OF EACH TONE IS "AMPCW"
C
      DO 10 I=1,NBD
      FREQ=F1CW-FXCW
      TAU(1)=TAU(1)+TS
      IF(TAU(1).GE.1.0)TAU(1)=TAU(1)-1.
      DO 5 J=1,NCW
      FREQ=FREQ+FXCW
      W=TWOPI*FREQ
    5 A(I)=A(I)+AMPCW*COS(W*TAU(1))
   10 CONTINUE
C
C                                   BINARY FSK
    2 M=KINTF.AND.2
      IF(M.EQ.0)GO TO 3
C                             "NFSK" TONE PAIRS
C                             FSK SHIFT="SHIFT" HZ
C                             SPACING BETWEEN ADJACENT PAIRS IS
C                             "FXFSK" HZ, PEAK AMPL OF EACH TONE IS
C                             "AMPFSK", RANDOM KEYING
      FREQ=F1MK-FXFSK
      DO 60 I=1,NFSK
      FREQ=FREQ+FXFSK
      FFSK(I)=FREQ
      X=RAN(I1,I2)
      IF(X.GT.0.5)FFSK(I)=FFSK(I)+SHIFT
   60 CONTINUE
      DO 20 I=1,NBD
      FREQ=F1MK-FXFSK
      TAU(2)=TAU(2)+TS
      IF(TAU(2).GE.1.)TAU(2)=TAU(2)-1.
      DO 15 J=1,NFSK
      W=TWOPI*FFSK(J)
   15 A(I)=A(I)+AMPFSK*COS(W*TAU(2))
   20 CONTINUE
C
    3 M=KINTF.AND.4
      IF(M.EQ.0)GO TO 4
C                             DPSK, 4PHASE
C                             "NPSK" TONES SPACED "FXPSK" HZ APART,
```

```
C                                           PEAK AMPL OF EACH TONE IS "AMPPSK"
C                                           RANDOM KEYING
      DO 70 I=1,NPSK
      L=1
      X=RAN(I1,I2)
      IF(X.GT.0.5)L=2
      X=RAN(I1,I2)
      IF(X.GT.0.5)L=L+1
      PHASE(I)=PHASE(I)+GRAY(L)*PI/4.0
      CALL LIMIT(PHASE(I))
   70 CONTINUE
      DO 30 I=1,NBD
      FREQ=F1PSK-FXPSK
      TAU(3)=TAU(3)+TS
      IF(TAU(3).GE.1.0)TAU(3)=TAU(3)-1.0
      DO 25 J=1,NPSK
      FREQ=FREQ+FXPSK
      W=TWOPI*FREQ
   25 A(I)=A(I)+AMPPSK*COS(W*TAU(3)+PHASE(J))
   30 CONTINUE
C
    4 CONTINUE
      M=KINTF.AND.8
      IF(M.EQ.0)RETURN
C                                           SWEPT TONE FQSAW BETWEEN F1SAW AND F2SAW
C                                                  SWEEP RATE IS RSAW
C                                                  AMPL IS AMPSAW
      DO 40 I=1,NBD
      FQSAW=FQSAW+RSAW/SAMP
      IF(FQSAW.GT.F2SAW)FQSAW=F1SAW
      TAU(4)=TAU(4)+TS
      IF(TAU(4).GE.1.)TAU(4)=TAU(4)-1.
      W=TWOPI*FQSAW
   40 A(I)=A(I)+AMPSAW*COS(W*TAU(4))
      RETURN
      END
C
C                                           ADD NOISE PULSE TO SIGNAL
      SUBROUTINE PULSE
      COMMON/BLK1/VAR(10,10),A(100),Q(100)
      COMMON/BLK6/I1,I2
      COMMON/BLK7/FLAG,KFLAG
      EQUIVALENCE (VAR(1,1),SAMP),(VAR(1,10),PROB)
      EQUIVALENCE (VAR(2,10),AMPPUL),(VAR(3,10),PULDUR)
      KPULSE=VAR(7,1)
      IF(KPULSE.EQ.0)RETURN
C-----------------------------------------------------------------
C
C     DUMMY SUBROUTINE
C
C     INPUT SIGNAL IS REAL IN ARRAY A
C     OUTPUT "         "  "    "   "   "
C     AMPPUL= RMS AMPLITUDE OF NOISE PULSE
C     PULDUR= DURATION OF NOISE PULSE
```

```
C       PROB= PROBABILITY OF OCCURRENCE OF NOISE PULSE
C       FLAG= CONTROL VARIABLE FOR NOISE PULSE
C       SAMP= SAMPLING RATE
C       NBD= NO. OF SAMPLES PROCESSED PER BLOCK
C       I1,I2= RANDOM NUMBER GENERATOR VARIABLES
C
C-----------------------------------------------------------------
        RETURN
        END
C
C                                FILTER SIGNAL+NOISE
        SUBROUTINE RXFILT
        COMMON/BLK1/VAR(10,10),A(100),Q(100)
        COMMON/BLK10/RXFIL(200)
        K=VAR(9,1)
        IF(K.LE.0)RETURN
C                       IF K=1, BASIC FILTER IS .5Z0-.5Z2
C                       IF K=2, ITERATE FILTER 1 TWO TIMES
C                       IF K=3, FILTER IS .6Z0-.5Z2-.1Z4
C                       IF K=4,ITERATE FILTER 3 TWO TIMES
C                       IF K=5, ITERATE FILTER 3 FOUR TIMES
        NBD=VAR(2,1)
        IF(K.GT.2)GO TO 9
C                       BASIC FILTER IS .5Z0-.5Z2
C                       ITERATE K TIMES
        DO 8 I=1,K
        DO 2 J=1,NBD
    2   RXFIL(J+2)=A(J)
        DO 4 J=1,NBD
    4   A(J)=.5*RXFIL(J+2)-.5*RXFIL(J)
        DO 5 J=1,2
    5   RXFIL(J)=RXFIL(J+NBD)
    8   CONTINUE
        RETURN
    9   K=K-2
        IF(K.GT.2)K=4
C                                BASIC FILTER IS .6Z0-.5Z2-.1Z4
C                                ITERATE FILTER K TIMES
        DO 100 I=1,K
        DO 10 J=1,NBD
   10   RXFIL(J+4)=A(J)
        DO 20 J=1,NBD
   20   A(J)=0.6*RXFIL(J+4)-0.5*RXFIL(J+2)-0.1*RXFIL(J)
        DO 30 J=1,4
   30   RXFIL(J)=RXFIL(J+NBD)
  100   CONTINUE
        RETURN
        END

C                       TX FILTER
        SUBROUTINE TXFILT
        COMMON/BLK1/VAR(10,10),A(100),Q(100)
        COMMON/BLK2/TXFIL(200)
C
```

```
      K=VAR(8,1)
      IF(K.EQ.0)RETURN
C                                       BASIC FILTER IS .6Z0-.5Z2-.1Z4
C                                       ITERATE K TIMES
      NBD=VAR(2,1)
      DO 100 I=1,K
      DO 10 J=1,NBD
   10 TXFIL(J+4)=A(J)
      DO 20 J=1,NBD
   20 A(J)=0.6*TXFIL(J+4)-0.5*TXFIL(J+2)-0.1*TXFIL(J)
      DO 30 J=1,4
   30 TXFIL(J)=TXFIL(J+NBD)
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE MTOUT(LUN,IDRIVE,ICNT,IARRAY,ISTAT)
C     MTOUT MAG-TAPE DRIVER
C     D. TATE   ORI,INC   JAN 79
C     THIS SUBROUTINE IS USED TO WRITE TO MAG TAPE
C     (NON-ANSI). VARIABLES ARE DEFINED IN I/O DRIVERS
C     REFERENCE MANUAL, SUBROUTINES ARE DEFINED IN
C     EXECUTIVE REFERENCE MANUAL.
C     PARAMETERS:
C     LUN = LOGICAL UNIT NUMBER (POSITION IN LOGICAL
C             UNIT TABLE)
C     IDRIVE = DRIVE NUMBER (0 OR 1)
C     ICNT < 0 => REWIND REQUEST
C          = 0 => WRITE EOF REQUEST
C          > 0 => NUMBER IF INTEGER WORDS IN RECORD
C     IARRAY = ARRAY TO TRANSFER TO TAPE
C     ISTAT = STATUS RETURNED TO CALLING PROGRAM
C            = 0 => O.K.
C            = 1 => WRITE PROTECTED (NO RING)
C            = 2 => END OF TAPE DETECTED
C            < 0 => UNSPECIFIED QIO$ ERROR
C
      DIMENSION ISB(2),IPRL(6),IARRAY(1)
      DATA ISSUC/1/,IOATT/"1400/,IEDAA/-8/IORWD/"2400/
      DATA IOEOF/"3000/,IOWLB/"400/,IEWLK/-12/,IEEOT/-62/
C ASSIGN LUN TO MAG TAPE UNIT
      CALL ASNLUN(LUN,'MT',IDRIVE,ISTAT)
      IF(ISTAT.NE.ISSUC) GOTO 99
C CLEAR PARAMETER LIST
      DO 10 I=1,6
10    IPRL(I)=0
C REQUEST EXCLUSIVE USE OF TAPE DRIVE
      CALL QIO(IOATT,LUN,1,0,ISB,IPRL,ISTAT)
C SEE IF QIO DIRECTIVE ACCEPTED
      IF(ISTAT.NE.ISSUC) GOTO 99
C QIO DIRECTIVE ACCEPTED, NOW WAIT FOR I/O TO FINISH
      CALL WAITFR(1,ISTAT)
C CHECK WAITFR DIRECTIVE ACCEPTANCE
      IF(ISTAT.NE.ISSUC) GOTO 99
C DIRECTIVE ACCEPTED, NOW CHECK I/O STATUS
```

```
        ISTAT=ISB(1)
        IF(ISB(1).EQ.ISSUC) GOTO 20
C IF DEVICE ALREADY ATTACHED, CONTINUE
C SIGN EXTEND FOR ERROR CODES
        ISTS=ISTAT-256
        IF(ISTS.NE.IEDAA) GOTO 99
C CHECK ICNT FOR FUNCTION
20      IF(ICNT.GE.0) GOTO 30
C REWIND REQUEST
        CALL QIO(IORWD,LUN,2,0,ISB,IPRL,ISTAT)
        GOTO 80
30      IF(ICNT.GT.0) GOTO 40
C WRITE EOF REQUEST
        CALL QIO(IOEOF,LUN,2,0,ISB,IPRL,ISTAT)
        GOTO 80
C NORMAL WRITE RECORD REQUEST
40      CALL GETADR(IPRL(1),IARRAY(1))
        IPRL(2)=ICNT*2
        CALL QIO(IOWLB,LUN,2,0,ISB,IPRL,ISTAT)
C CHECK DIRECTIVE ACCEPTANCE FOR ALL FUNCTIONS
80      IF(ISTAT.NE.ISSUC) GOTO 99
        CALL WAITFR(2,ISTAT)
        IF(ISTAT.NE.ISSUC) GOTO 99
C FUNCTION WAS ACCEPTTED, NOW CHECK FINAL STATUS
        ISTAT=0
        IF(ISB(1).EQ.ISSUC) GOTO 999
C SIGN EXTEND STATUS FOR ERROR CODES
        ISTS=ISB(1)-256
        IF(ISTS.EQ.IEWLK) ISTAT=257
        IF(ISTS.EQ.IEEOT) ISTAT=258
C NOT LOCKED OR EOT, BUT STILL AN ERROR
        IF(ISTAT.EQ.0) ISTAT=ISB(1)
C       ERROR RETURN
99      ISTAT=ISTAT-256
C       NORMAL RETURN
999     RETURN
        END
C       LIMIT THE ANGLE P  TO +-PI
        SUBROUTINE LIMIT(P)
        IF(P.LT.0.0)GO TO 100
C   P IS POSITIVE, LIMIT TO +-TWOPI
        DO 10 I=1,500
        IF(P.LT.6.2831853072)GO TO 20
   10   P=P-6.2831853072
C    LIMIT P  +-PI
   20   IF(P.GT.3.1415926536)P=P-6.2831853072
        GO TO 300
C    P IS NEGATIVE,  LIMIT TO -TWOPI
  100   DO 150 I=1,500
        IF(P.GT.-6.2831853072)GO TO 160
  150   P=P+6.2831853072
C  LIMIT TO +-PI
  160   IF(P.LT.-3.1415926536)P=P+6.2831853072
  300   RETURN
        END
```

```
C                 RECEIVER AUTOMATIC GAIN CONTROL
C
        SUBROUTINE AGC
        COMMON/BLK1/VAR(10,10),A(100),O(100)
        COMMON/BLK11/AGCFIL(2,100)
        EQUIVALENCE (VAR(2,2),GAIN),(VAR(6,2),DECAY),(VAR(7,2),AGCMIN)
C
        KAGC=VAR(10,1)
        IF(KAGC.EQ.0)RETURN
        NBD=VAR(2,1)
        NAGC=VAR(5,2)
C
C-------------------------------------------------
C
C       DUMMY SUBROUTINE
C
C       GAIN= CURRENT GAIN VALUE
C       NAGC= ATTACK CONSTANT
C       DECAY= DECAY CONSTANT
C       AGCMIN= CONTROLS MAX AMPLIFICATION
C
C-------------------------------------------------
        RETURN
        END
        SUBROUTINE MTSET(LUN,IDRIVE,ISET,ISTAT)
C       MTSET MAG TAPE DRIVER (SET CHARACTERISTICS)
C       D. TATE   ORI,INC   AUG 80
C       THIS SUBROUTINE SETS THE TAPE CHARACTERISTICS. IT
C       DOES NOT ATTACH THE UNIT. VARIABLES ARE DEFINED IN
C       THE I/O DRIVERS REFERENCE MANUAL, SUBROUTINES ARE DEFINED
C       IN THE EXECUTIVE REFERENCE MANUAL.
C
C       PARAMETERS:
C         LUN - LOGICAL UNIT NUMBER
C         IDRIVE - MAG TAPE DRIVE NUMBER (0 OR 1)
C         ISET - 0 = SET DRIVE FOR NORMAL REWRITE CAPABILITY
C                1 = SET DRIVE FOR NO REWRITE CAPABILITY
C               NOTE: ISET SHOULD BE SET TO 1 IF THE TAPE IS TO BE
C                       PLAYED IN THE ANALOG MODE. THE ANALOG
C                       CONTROLLER CAN NOT TOLERATE EXTENDED INTER-
C                       RECORD GAPS. MTOUT WRITE ERRORS WILL RETURN
C                       WITH ISTAT = -56 (IE.BBE - BAD BLOCK)
C         ISTAT - ERROR STATUS RETURNED TO CALLING PROGRAM
C                       (0 = NO ERROR)
C
        DIMENSION IPRL(6),ISB(2)
        DATA IPRL/6*0/,ISB/2*0/
        DATA IOSTC/"2500/,ISSUC/1/
C
C ASSIGN LUN TO MAG TAPE UNIT
        CALL ASNLUN(LUN,'MT',IDRIVE,ISTAT)
        IF(ISTAT.NE.ISSUC) GOTO 99
C DETERMINE REQUESTED CHARACTERISTIC
```

```
      IPRL(1)=0
      IF(ISET.NE.0) IPRL(1)=128
C SET TAPE CHARACTERISTICS
      CALL QIO(IOSTC,LUN,1,0,ISB,IPRL,ISTAT)
C SEE IF QIO DIRECTIVE ACCEPTED
      IF(ISTAT.NE.ISSUC) GOTO 99
C QIO DIRECTIVE ACCEPTED, NOW WAIT FOR I/O TO FINISH
      CALL WAITFR(1,ISTAT)
C CHECK WAITFR DIRECTIVE ACCEPTANCE
      IF(ISTAT.NE.ISSUC) GOTO 99
C DIRECTIVE ACCEPTED, CHECK I/O STATUS
      IF(ISB(1).NE.ISSUC) GOTO 99
      ISTAT=0
99    RETURN
      END
C
      SUBROUTINE STOREX
C                       STORE DATA ON MT: OR DISK
      COMMON/BLK1/VAR(10,10),A(100),Q(100)
      COMMON/BLK12/IB(1000),JSTAT,NDZ,KCODE,RMS,PEAK,TOTAL
      NBD=VAR(2,1)
      MTAPE=VAR(4,2)
      DO 100 I=1,NBD
C                       REMOVE NORMALIZATION
C                       ACCUMULATE RMS, PEAK VALUE, TOTAL NO.
C                       SAMPLES
      A(I)=A(I)*317.12
      RMS=RMS+A(I)*A(I)
      IF(A(I).GT.PEAK)PEAK=A(I)
      TOTAL=TOTAL+1.
      NDZ=NDZ+1
C                       CONVERT TO INTEGER, ADD BLOCKER CODE
      IB(NDZ)=IFIX(A(I))/16+KBCODE
      KBCODE=KBCODE+4
      IF(KBCODE.GT.15)KBCODE=3
      IF(NDZ.LT.1000)GO TO 100
      NDZ=0
      IF(MTAPE.GT.1)WRITE(2)IB
      IF(MTAPE.LE.1)CALL MTOUT(2,MTAPE,1000,IB,JSTAT)
C               CLEAR INTEGER ARRAY
      DO 50 J=1,1000
   50 IB(J)=0
  100 CONTINUE
      RETURN
      END
```

```
C                         APPENDIX B
C
C                         PROGRAM VARGEN
C                PROGRAM TO GENERATE TABLE OF VARIABLES FOR STRESS PROGRAM
C        ALL ENTRIES ARE FROM KEYBOARD IN F9.3 FORMAT IN RESPONSE
C                TO STATEMENTS
C        ALL DATA IS STORED IN IN ARRAY VAR(10,10)
C
         DIMENSION VAR(10,10)
         WRITE(5,1)
 1       FORMAT(1X,'ENTER SAMP, BAUD, PATHS,FADING',/)
         DO 100 I=1,4
  100    READ(5,2)VAR(I,1)
    2    FORMAT(F9.3)
         WRITE(5,3)
    3    FORMAT(1X,'ENTER TYPES- DOP, INTERF, PULSE, FILTERS(TX,RX)',/)
         DO 200 I=5,9
  200    READ(5,2)VAR(I,1)
         WRITE(5,20)
   20    FORMAT(1X,'ENTER AGC TYPE, GAUSSIAN NOISE LEVEL',/)
         READ(5,2)VAR(10,1)
         READ(5,2)VAR(1,2)
         WRITE(5,66)
   66    FORMAT(1X,'ENTER AGC GAIN, AGC ATTACK, AGC DECAY, AGCMAX',/)
         READ(5,2)VAR(2,2)
         READ(5,2)VAR(5,2)
         READ(5,2)VAR(6,2)
         READ(5,2)VAR(7,2)
         WRITE(5,67)
   67    FORMAT(1X,'ENTER INPUT, OUTPUT DEVICE',/)
         READ(5,2)VAR(3,2)
         READ(5,2)VAR(4,2)
         N=VAR(3,1)
         WRITE(5,4)N
    4    FORMAT(1X,'ENTER DELAY TIMES FOR ',I2,' PATHS',/)
         DO 300 I=1,N
  300    READ(5,2)VAR(I,3)
         WRITE(5,6)N
    6    FORMAT(1X,'ENTER DELAY AMP FOR ',I2,' PATHS',/)
         DO 500 I=1,N
  500    READ(5,2)VAR(I,4)
         K=VAR(6,1)
         M=K.AND.1
         IF(M.EQ.0)GO TO 22
         WRITE(5,7)
    7    FORMAT(1X,'ENTER CW  NTONE,LOWEST, SPACING, AMPL',/)
         DO 600 I=1,4
  600    READ(5,2)VAR(I,5)
   22    M=K.AND.2
         IF(M.EQ.0)GO TO 24
         WRITE(5,8)
    8    FORMAT(1X,'ENTER FSK NTONE,F1MARK,SHIFT,SPACING,AMPL',/)
```

```
      DO 1600 I=1,5
1600  READ(5,2)VAR(I,6)
 24   M=K.AND.4
      IF(M.EQ.0)GO TO 26
      WRITE(5,10)
 10   FORMAT(1X,'ENTER DPSK NTONE,F1,SPACING,AMPL',/)
      DO 700 I=1,4
700   READ(5,2)VAR(I,7)
 26   M=K.AND.8
      IF(M.EQ.0)GO TO 30
      WRITE(5,27)
 27   FORMAT(1X,'ENTER SWEPT TONE- LOW,HIGH,RATE,AMPL',/)
      DO 28 I=1,4
 28   READ(5,2)VAR(I,8)
 30   K=VAR(5,1)
      IF(K.EQ.0)GO TO 40

      WRITE(5,11)
 11   FORMAT(1X,'ENTER START TIME FOR DOPPLER CHANGE',/)
      READ(5,2)VAR(1,9)
      IF(K.GT.1)GO TO 35
      WRITE(5,12)
 12   FORMAT(1X,'ENTER DOPPLER STEP SIZE',/)
      READ(5,2)VAR(2,9)
      GO TO 40
 35   IF(K.GT.2)GO TO 38
      WRITE(5,36)
 36   FORMAT(1X,'ENTER DOPP RAMP RATE',/)
      READ(5,2)VAR(3,9)
      WRITE(5,13)
 13   FORMAT(1X,'ENTER DOPP LOW AND HIGH LIMIT',/)
      READ(5,2)VAR(4,9)
      READ(5,2)VAR(5,9)
      GO TO 40
 38   CONTINUE
      WRITE(5,14)
 14   FORMAT(1X,'ENTER FLYBY START TIME AND MIN DISTANCE',/)
      READ(5,2)VAR(6,9)
      READ(5,2)VAR(7,9)
      WRITE(5,15)
 15   FORMAT(1X,'ENTER FLYBY VELOCITY AND RADIO FREQ',/)
      READ(5,2)VAR(8,9)
      READ(5,2)VAR(9,9)
 40   WRITE(5,41)
 41   FORMAT(1X,'ENTER FREQ TRANSLATION ERROR',/)
      READ(5,2)VAR(10,9)
      IF(VAR(7,1).EQ.0.)GO TO 50
      WRITE(5,16)
 16   FORMAT(1X,'ENTER PULSE PROB,AMPL,DURATION',/)
      DO 17 I=1,3
 17   READ(5,2)VAR(I,10)
 50   CONTINUE
      CALL ASSIGN(2,'STRVAR.DAT')
      WRITE(2)VAR
      ENDFILE 2

      CALL CLOSE(2)
      END
```

```
C                                APPENDIX C
C                                PROGRAM EDVAR
C
C       EDIT TABLE OF VARIABLES FOR STRESS PROGRAM
C
C       USER SPECIFIES LOCATION TO BE CHANGED AND NEW VALUE
C       LOCATION IDENTIFIED BY FORMAT I2,1X,I2
C       TERMINATE BY ENTERING ZERO
C       NEW VALUE ENTERED BY FORMAT F9.3
C       PROGRAM IDENTIFIES OLD VALUE
C
C               DATA STORED IN DISK FILE STRVAR.DAT
C
        DIMENSION VAR(10,10)
        CALL ASSIGN(2,'STRVAR.DAT')
        READ(2,END=3000,ERR=3000)VAR
        CALL CLOSE(2)
   10   WRITE(5,1)
    1   FORMAT(1X,'ENTER LOCATION',/)
        READ(5,2)I,J
    2   FORMAT(I2,1X,I2)
        IF(I.EQ.0)GO TO 30
        WRITE(5,3)VAR(I,J)
    3   FORMAT(1X,F9.3)
        WRITE(5,4)
    4   FORMAT(1X,'ENTER CHANGE',/)
        READ(5,5)VAR(I,J)
    5   FORMAT(F9.3)
        GO TO 10
   30   CALL ASSIGN(2,'STRVAR.DAT')
        WRITE(2)VAR
        ENDFILE 2
        CALL CLOSE(2)
 3000   CONTINUE
        END
```

```
C             APPENDIX D
C
C             PROGRAM PATHM
C       GENERATE A FILE OF COMPLEX VALUES EQUAL TO THE SUM
C       OF NTONE SINUSOIDS OF DIFFERENT FREQUENCY, REPRESENTING THE
C       COMPLEX GAIN OVER ONE PATH
C       MAXIMUM OF 10 PATHS, MAXIMUM OF 30 TONES/PATH
C       DO THIS FOR M PATHS, INTERLEAVE RESULTS
C       NTONE FREQUENCIES FOR EACH PATH ARE RANDOMLY SELECTED
C       WITHIN NTONE INCREMENTS OVER TOTAL SPREAD
C       STORE IN BLOCKS OF 1024 COMPLEX VALUES
C
C       INPUTS AT RUN TIME ARE:
C             MODEM SAMPLING RATE
C             NO. OF PATHS
C             DOPPLER SPREAD AND MEAN FREQUENCY OFFSET FOR EACH PATH
C             NO. OF TONES USED TO GENERATE GAIN VALUES
C             NO. OF SECONDS OF DATA
C
C       OUTPUT STORED IN DISK FILE PATH.DAT
C
        DIMENSION WT(10,30),F(10,30)
        DIMENSION SPR(10),DOP(10)
        COMPLEX W(1024)
        WRITE(5,1)
    1   FORMAT(1X,'ENTER MODEM SAMPLING FREQ',/)
        READ(5,2)SAMP
C                           TAP GAIN VALUES GENERATED AT 0.01*SAMP
        SAMP=SAMP/100.
    2   FORMAT(F9.2)
        WRITE(5,3)
    3   FORMAT(1X,'ENTER NO. OF TONES/PATH',/)
        READ(5,4)NTONE
    4   FORMAT(I5)
        WRITE(5,5)
    5   FORMAT(1X,'ENTER NO. OF PATHS, AND SPREAD AND DOP FOR EACH',/)
        READ(5,4)NPATH
        DO 10 I=1,NPATH
        WRITE(5,6)
    6   FORMAT(1X,'ENTER SPREAD',/)
        READ(5,7)SPR(I)
        WRITE(5,8)
    8   FORMAT(1X,'ENTER DOPPLER',/)
        READ(5,7)DOP(I)
    7   FORMAT(F9.6)
   10   CONTINUE
        WRITE(5,20)
   20   FORMAT(1X,'ENTER NO.OF SECONDS OF DATA TO GENERATE',/)
        READ(5,2)SEC
                        CONVERT SECOND TO BLOCKS OF 1024
        NBLK=1+SEC*SAMP*NPATH/1024.
        I1=0
```

```
      I2=0
      TWOPI=6.28318
      DO 30 I=1,100
   30 X=RAN(I1,I2)
C                          GENERATE NTONE RANDOM FREQUENCIES FOR EACH PATH
      DO 40 I=1,NPATH
      FX=2.*SPR(I)/NTONE
      FY=-SPR(I)
      DO 35 J=1,NTONE
      X=RAN(I1,I2)
      F(I,J)=FY+X*FX+DOP(I)
      X=RAN(I1,I2)
      WT(I,J)=X*TWOPI
      WRITE(5,32)I,J,F(I,J)
   32 FORMAT(1X,2I5,2X,F8.4)
      F(I,J)=F(I,J)*TWOPI/SAMP
   35 FY=FY+FX
   40 CONTINUE
      DO 50 I=1,NPATH
      DO 45 J=1,NTONE
   41 IF(ABS(F(I,J)).LT.TWOPI)GO TO 45
      IF(F(I,J).GE.TWOPI)F(I,J)=F(I,J)-TWOPI
      IF(F(I,J).LE.-TWOPI)F(I,J)=F(I,J)+TWOPI
      GO TO 41
   45 CONTINUE
   50 CONTINUE
      CALL ASSIGN(3,'PATH.DAT')
      NDX=0
      JBLK=0
      SQNT=SQRT(FLOAT(NTONE))
C                              START OF MAIN LOOP
C                              TO GENERATE GAIN VALUES,
C                              FOR EACH PATH GENERATE AN I AND Q VALUE
  100 CONTINUE
      DO 1000 I=1,NPATH
      A=0.
      Q=0.
      DO 400 J=1,NTONE
      WT(I,J)=WT(I,J)+F(I,J)
      IF(WT(I,J).GE.TWOPI)WT(I,J)=WT(I,J)-TWOPI
      IF(WT(I,J).LE.-TWOPI)WT(I,J)=WT(I,J)+TWOPI
      A=A+COS(WT(I,J))
  400 Q=Q+SIN(WT(I,J))
C                                        NORMALIZE
      A=A/SQNT
      Q=Q/SQNT
C                                        STORE
      NDX=NDX+1
      W(NDX)=CMPLX(A,Q)
      IF(NDX.LT.1024)GO TO 1000
      NDX=0
      WRITE(3)W
C                                        INCREMENT BLOCK COUNT
      JBLK=JBLK+1
 1000 CONTINUE

C                                        TEST BLOCK COUNT
      IF(JBLK.LT.NBLK)GO TO 100
      ENDFILE 3
      CALL CLOSE(3)
      END
```

DATE

ILME